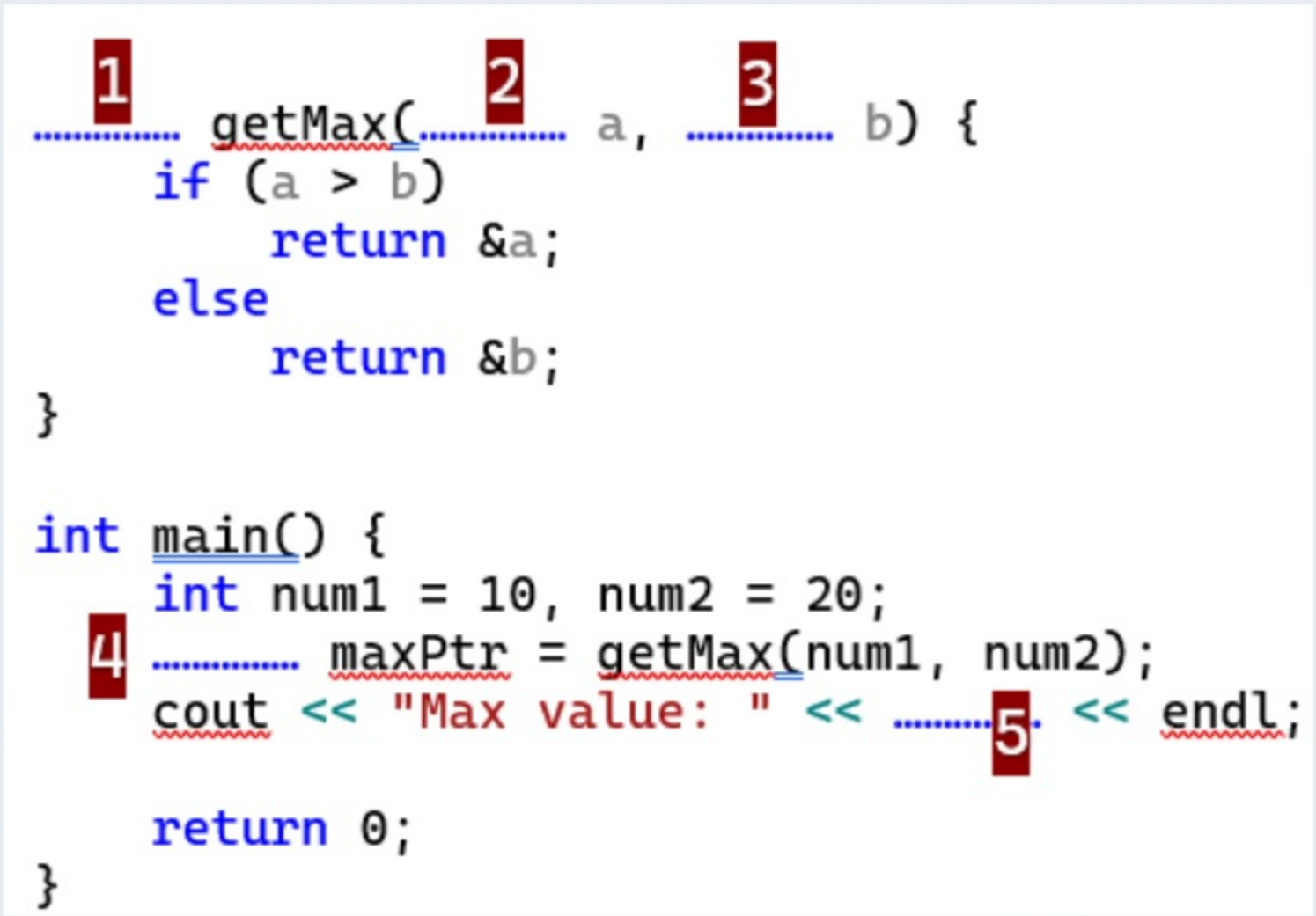
* Required

Question #1

Complete the following code to return the max of two numbers ?

* (5 Points)



- [x] **int*, int&, int&, int*, *maxPtr**
- [ ] **&int, int, int, int, maxPtr**
- [ ] **int&, int*, int*, int, *maxPtr**
- [ ] **int*, int, int, int, maxPtr**
- [ ] ***int, int&, int&, int, *maxPtr**

Question #2

Complete the following code to insert an element at specific position in static array

* (5 Points)

```
void Insert(int element, int index) {
        if (.....1..... >= .....2.....) {
                throw std::overflow_error("Array is Full...");
        }
        for (size_t i = .....3.....; i >= .....4.....; i--)
        {
                arr[i] = arr[.....5.....];
        }
        arr[index] = element;
        arrsize++;
}
```

- ( ) capacity, arrsize, arrsize, index, i+1
- ( ) arrsize, capacity, index, arrsize, i-1
- ( ) capacity, arrsize, index, arrsize, i
- (•) arrsize, capacity, arrsize, index, (i - 1)
- ( ) arrsize, capacity, index, arrsize-1,(i-1)

Back Next

Question #3

Complete the following code to resize the capacity of dynamic array

* (5 Points)

```
int* data;              // Pointer to array data
size_t capacity;
size_t size;

void resize(size_t newCapacity) {
    1 ..... newData = new int[newCapacity];
    for (size_t i = 0; i < size; ++i) {
        newData[i] = ....2.... [i];
    }
    ....3.... data;
    data = .....4;
    capacity = .....5...;
}
```

- ○ T*, size, delete,newCapacity,newData
- ◉ int*, data, delete[], newData, newCapacity
- ○ int&, *data, delete[], newCapacity, newData
- ○ T*, *data, delete, newData, (newCapacity + 1)
- ○ int*, &data, delete[], newData, newCapacity

Back Next

Question #4

-Based on linked list insertion operation, answer the following questions

```
//Insert at end (Append)
void insertatend(int value) {
    1 Node* newnode = new Node(value);
      if (head == nullptr)2{
            head = newnode; 3
      }
      else {
            Node* temp = head;
            while (.....4.....) {
                  temp = temp->next;
            }
            temp->next = newnode;5
      }
}
```

Q:4.1) Statement #1 means ..... * (1 Point)

- ○ Garbage Memory
- ○ Memory deallocation
- ○ Memory Leak
- ◉ Memory Allocation
- ○ dangle pointer creation

Q:4.2) Condition in Line Statement #2 check if ..... *
(1 Point)

- ◉ Linked list is empty
- ○ head pointer is deleted
- ○ Linked list is full
- ○ head pointer not founded
- ○ head pointer = tail pointer

Q:4.3) If the condition in Statement #2 is true, Statement #3 means ..... * (1 Point)

- ( ) Linked list is full capacity
- ( ) head pointer refers to last node
- ( ) head pointer will store newnode address then deleted
- ( ) move head pointer one step forward to the newnode
- (•) newnode will be the first node

Q:4.4) Complete the condition on Statement #4 * (1 Point)

- ( ) head->data!= nullptr
- (•) temp->next != nullptr
- ( ) True
- ( ) head->next != nullptr
- ( ) temp

Q:4.5) Statement #5 means ..... * (1 Point)

- (•) Update pointer
- ( ) move pointer forward
- ( ) move pointer backward
- ( ) dereference newnode pointer
- ( ) delete second last node